s0ybtbiib

August 22, 2024

1 Model Selection Practice

Aug 22, 2024

1.1 Bài toán

- Cần đánh giá hiệu quả của một mô hình phân loại?
- So sánh hiệu quả của 2 mô hình khác nhau?

Nhưng ta chỉ có một tập dữ liệu đã thu thập được. Để trả lời hai câu hỏi trên thì cần thực hiện bước "Lựa chọn tham số" của mô hình đã chọn.

Bài này sẽ hướng dẫn cách thực hiện từng bước chi tiết, từ lựa chọn tham số (sử dụng Cross validation), cho đến đánh giá (sử dụng Holdout) và so sánh hai mô hình khác nhau.

====== Nguồn http://users.soict.hust.edu.vn/khoattq/ml-dm-course/ ======

Mục tiêu:

- Nắm được các bước đánh giá và so sánh hai mô hình khác nhau (*ví dụ: SVM và Random Forest*).
- Ứng dụng vào giải quyết bài toán thực tế (*ví dụ: phân loại văn bản*).
- Sử dụng độ đo Accuracy để đánh giá chất lượng mô hình.

Dữ liệu: * Tập các văn bản và nhãn tương ứng của từng văn bản trong một khoảng thời gian. * Tập các nhãn - 10 nhãn văn bản:

> Giải trí, Khoa học - Công nghệ, Kinh tế, Pháp luật, Sức khỏe, Thể thao, Thời sự, Tin khác, Độc giả, Đời sống - Xã hội.

Ví dụ văn bản nhãn thể thao:

> "Dân_trí Real Madrid đã dẫn trước trong cả trận đấu , nhưng họ vẫn phải chấp_nhận bị Dortmund cầm hòa 2-2 ở Bernabeu . Real Madrid chấp_nhận đứng thứ_hai ở bảng F Champions League …"

Bài toán: Phân loại

- Input: n vector mã hóa của các văn bản - ma trận $X = [x_1, x_2, ...x_n]$
- Output: nhãn y là 1 trong 10 nhãn trên

[ ]:
```
%cd /content/drive/MyDrive/Code_VinBigData_2024
```

```
[ ]: os.makedirs("Model_selection/images",exist_ok=True)  # thư mục lưu các hình ảnh
      ↪kết quả trong quá trình huấn luyện và đánh giá
```

1.2 Import các thư viện cần thiết, cài thêm một số thư viện chưa sẵn có

```
[ ]: !pip install pyvi
```

```
[ ]: import os
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import learning_curve
from tqdm import tqdm

from sklearn.datasets import load_files
from pyvi import ViTokenizer

from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix,
      ↪ConfusionMatrixDisplay

%matplotlib inline
```

1.3 Load dữ liệu từ thư mục đã crawl từ trước

Cấu trúc thư mục như sau: - data/news_1135/ - Kinh tế/ - bài báo 1.txt - bài báo 2.txt - Pháp luật/ - bài báo 3.txt - bài báo 4.txt

```
[ ]: INPUT = "data/data/news_1135/"

### bài tập ###
# yêu cầu: In ra các nhãn và số văn bản tương ứng trong mỗi nhãn; Tổng số văn
      ↪bản
# gợi ý: tự làm
################
# code

################
```

1.4 Tiền xử lý dữ liệu đưa dữ liệu từ dạng text về dạng ma trận

[ ]:
```
### bài tập ###
# yêu cầu: Tiền xử lý dữ liệu text về dạng ma trận, sử dụng TF-IDF.
# gợi ý: tự làm.
###############
# code

###############
print(f"Số lượng stopwords: {len(stopwords)}")
print(stopwords[:10])
print(f"\nSố lượng từ trong từ điển: {len(module_count_vector.vocabulary_)}")
print(f"Kích thước dữ liệu sau khi xử lý: {data_preprocessed.shape}")
print(f"Kích thước nhãn tương ứng: {data_train.target.shape}")
```

1.5 Chia dữ liệu làm 2 phần training và testing

- Training chiếm 80 % dữ liệu
- Testing chiếm 20 % dữ liệu

  Nghĩa là ta sẽ dùng **Holdout** để đánh giá hiệu quả của một mô hình

[ ]:
```
test_size = 0.2
# cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
X_train, X_test, y_train, y_test = train_test_split(data_preprocessed,␣
 ↪data_train.target, test_size=test_size, random_state=0)

print("Dữ liệu training = ", X_train.shape, y_train.shape)
print("Dữ liệu testing = ", X_test.shape, y_test.shape)
```

[ ]:
```
### bài tập ###
# yêu cầu: Không sử dụng train_test_split, chia dữ liệu thành hai tập Train,␣
 ↪Test tương ứng
# gợi ý: tự làm
###############
# code

###############
```

1.6 Lựa chọn (tối ưu) tham số

Chỉ dùng **tập train** để thực hiện **lựa chọn tham số**.

- SVM: kernel, C
- Random Forest: criteria, N

Ta sẽ dùng chiến lược Cross Validation trong bước này.

[ ]:
```
def cross_validation(estimator):
    _, train_scores, test_scores = learning_curve(estimator, X_train, y_train,␣
 ↪cv=5, n_jobs=-1, train_sizes=[1.0, ], scoring="accuracy")
    test_scores = test_scores[0]
    mean, std = test_scores.mean(), test_scores.std()
    return mean, std

def plot(title, xlabel, X, Y, error, ylabel = "Accuracy"):
    plt.xlabel(xlabel)
    plt.title(title)
    plt.grid()
    plt.ylabel(ylabel)

    plt.errorbar(X, Y, error, linestyle='None', marker='o')
```

1.7 Lựa chọn tham số mô hình SVM

[ ]:
```
Save_figs = "/content/drive/MyDrive/Code_VinBigData_2024/Model_selection/images/
 ↪"
```

1.7.1 Đánh giá ảnh hưởng của các kernel trong SVM

[ ]:
```
title = "thay đổi kernel, C = 1"
xlabel = "kernel"
X = []
Y = []
error = []

for kernel in tqdm(['linear', 'poly', 'rbf', 'sigmoid']):
    # Với mỗi kernel được chọn,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = svm.SVC(kernel=kernel, C=1.0)
    mean, std = cross_validation(text_clf)
    X.append(kernel)
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig(Save_figs+'svm_change_kernel.png', bbox_inches='tight')
plt.show()
```

1.7.2 Đánh giá ảnh hưởng của tham số C trong SVM

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của tham số C trong SVM, sử dụng kernel 'Linear'
#          Vẽ và Lưu biểu đồ kết quả với tên "svm_change_C.png"
# gợi ý: Xét giá trị C trong khoảng [.1, 1.0, 2.0, 5.0, 10.0]
################
# code

################
```

1.8 Lựa chọn tham số mô hình Random forest

1.8.1 Đánh giá ảnh hưởng của độ đo trong Random Forest

[ ]:
```
title = "thay đổi criterion, n_estimators = 50"
xlabel = "criterion"
X = []
Y = []
error = []

for criterion in tqdm(["gini", "entropy"]):
    # Với mỗi criterion nhận được,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = RandomForestClassifier(criterion=criterion, n_estimators=50)
    mean, std = cross_validation(text_clf)
    X.append(str(criterion))
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig(Save_figs+'RF_change_criterion.png', bbox_inches='tight')
plt.show()
```

1.8.2 Đánh giá ảnh hưởng của số cây trong Random Forest

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của số lượng cây trong RF, sử dụng criterion =␣
 ↪'gini'
#          Vẽ và Lưu biểu đồ kết quả với tên "RF_change_N.png"
# gợi ý: Xét giá trị C trong khoảng [10, 50, 100, 300]
################
# code

################
```

1.9 Lựa chọn tham số cho mô hình KNN

[ ]:
```
from sklearn.neighbors import KNeighborsClassifier

title = "thay đổi K"
xlabel = "K"
X = []
Y = []
error = []

for k in tqdm([1, 3, 5, 20, 50]):
    # Với từng giá trị k nhận được,
    # thực hiện xây dựng mô hình, huấn luyện và đánh giá theo cross-validation
    text_clf = KNeighborsClassifier(n_neighbors=k)
    mean, std = cross_validation(text_clf)
    X.append(str(k))
    Y.append(mean)
    error.append(std)

# lưu kết quả ra file ảnh
plot(title, xlabel, X, Y, error)
plt.savefig(Save_figs+'KNN_change_K.png', bbox_inches='tight')
plt.show()
```

1.10 So sánh các mô hình

- Sau khi chọn được các bộ tham số tốt nhất cho mỗi mô hình, ta huấn luyện lại trên toàn bộ tập Train.
- Dùng các mô hình mới huấn luyện để phán đoán cho các dữ liệu trong tập Test
- Đo đạc Độ chính xác (Accuracy) của chúng và so sánh kết quả.

[ ]:
```
svm_ = svm.SVC(kernel='linear', C=1.0)
rf = RandomForestClassifier(criterion='gini', n_estimators=300)
knn = KNeighborsClassifier(n_neighbors=5)

# Huấn luyện các mô hình trên tập dữ liệu train đầy đủ
svm_.fit(X_train, y_train)
rf.fit(X_train, y_train)
knn.fit(X_train, y_train)
```

[ ]:
```
# Kết quả dự đoán trên tập test
print(f'SVM: {accuracy_score(y_test, svm_.predict(X_test))}')
print(f'RF: {accuracy_score(y_test, rf.predict(X_test))}')
print(f'KNN: {accuracy_score(y_test, knn.predict(X_test))}')
```

[ ]:
```
Y_pred_svm = svm_.predict(X_test)
cm_svm = confusion_matrix(y_test, Y_pred_svm, labels=svm_.classes_)
```

```
disp_svm = ConfusionMatrixDisplay(confusion_matrix=cm_svm, display_labels=svm_.
 ↪classes_)
disp_svm.plot()
plt.show()
```

[ ]:
```
### bài tập ###
# yêu cầu: Vẽ biểu đồ Confusion_matrix của kết quả dự đoán trên tập Test của␣
 ↪hai mô hình RF và KNN
# gợi ý: tương tự cách làm với SVM
###############
# code

###############
```

1.11 Bài tập

- Sử dụng dữ liệu đánh giá tín dụng cá nhân
- Sử dụng độ đo đánh giá negative cost
- Lựa chọn tham số cho các mô hình SVM, Random Forest và KNN
- So sánh các mô hình với siêu tham số tốt nhất

1.11.1 Đọc dữ liệu

[ ]:
```
data = np.genfromtxt('/content/drive/MyDrive/Code_VinBigData_2024/
 ↪Model_selection/german.data-numeric')
X_data = data[:,:24]
Y_data = data[:,-1]
print(X_data.shape)
print(Y_data.shape)
```

1.11.2 Chia dữ liệu Train - Test

[ ]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.
 ↪2, random_state=42)
print("Dữ liệu training = ", X_train.shape, Y_train.shape)
print("Dữ liệu testing = ", X_test.shape, Y_test.shape)
```

1.11.3 Các hàm cần thiết

[ ]:
```
# Hàm tính neg_cost, dùng để truyền vào scoring của learning_curve
def neg_cost(estimator, X, y):
    y_true = y
    y_pred = estimator.predict(X)
    true_pos = ((y_true==y_pred)&(y_true==1.0))*0.0
    true_ne = ((y_true==y_pred)&(y_true==2.0))*0.0
```

```
    false_ne = ((y_true!=y_pred)&(y_true==1.0))*1.0
    false_pos = ((y_true!=y_pred)&(y_true==2.0))*5.0
    return -sum(true_pos + true_ne + false_pos + false_ne)/len(y_true)

def cross_validation(estimator):
    _, train_scores, test_scores = learning_curve(estimator, X_train, Y_train,␣
 ↪cv=10, n_jobs=-1, train_sizes=[0.8, ], scoring=neg_cost)
    test_scores = test_scores[0]
    mean, std = test_scores.mean(), test_scores.std()
    return mean, std

def plot(title, xlabel, X, Y, error, ylabel = "neg cost"):
    plt.xlabel(xlabel)
    plt.title(title)
    plt.grid()
    plt.ylabel(ylabel)

    plt.errorbar(X, Y, error, linestyle='None', marker='o')
```

1.11.4 SVM

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của các kernel trong SVM, chọn C=1
#          Vẽ và Lưu biểu đồ kết quả với tên "prac_svm_change_kernel.png"
# gợi ý: Xét các kernels: ['linear', 'poly', 'rbf', 'sigmoid']
################
# code

################
```

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của tham số C trong SVM, sử dụng kernel = 'linear'
#          Vẽ và Lưu biểu đồ kết quả với tên "prac_svm_change_C.png"
# gợi ý: Xét giá trị C trong khoảng [.1, 1.0, 2.0, 5.0, 10.0]
################
# code

################
```

1.11.5 Random Forest

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của độ đo trong Random Forest, chọn số cây = 50
#          Vẽ và Lưu biểu đồ kết quả với tên "prac_RF_change_criterion.png"
# gợi ý: Xét phương pháp ["gini", "entropy"]
################
# code
```

```
################
```

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của số cây trong Random Forest, sử dụng criterion␣
 ↪= 'entropy'
#          Vẽ và Lưu biểu đồ kết quả với tên "prac_RF_change_N.png"
# gợi ý: Xét giá trị Cây trong khoảng [10, 50, 100, 300]
################
# code

################
```

1.11.6 KNN

[ ]:
```
### bài tập ###
# yêu cầu: Đánh giá ảnh hưởng của số lượng láng giềng trong KNN
#          Vẽ và Lưu biểu đồ kết quả với tên "prac_KNN_change_K.png"
# gợi ý: Xét giá trị trong khoảng [1, 3, 5, 20, 50]
################
# code

################
```

1.11.7 So sánh

[ ]:
```
### bài tập ###
# yêu cầu: So sánh hiệu suất của 3 mô hình trên
# gợi ý: Thực hiện 2 bước:
#       - Huấn luyện lại mô hình với tập Train đầy đủ
#       - Dự đoán tập Test
################
# code

###Bước 1:
# Huấn luyện các mô hình trên tập dữ liệu train đầy đủ

###Bước 2:
# Kết quả dự đoán trên tập test
################
```